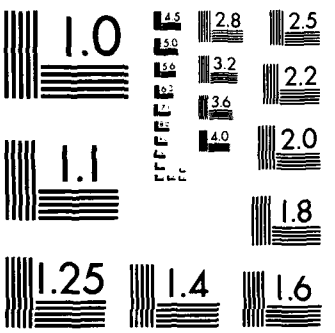
AD-A195 822 DOCUMENTATION OF SOURCE CODE(U) JAYCOR ALEXANDRIA VA 1/1
12 MAY 88 N00014-85-C-2444

UNCLASSIFIED F/G 12/5 NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1
8.0
9.0

FINAL REPORT

&

DOCUMENTATION OF SOURCE CODE

Deliverable Nos.: A002 & A003

Prepared by:

JAYCOR

Prepared for:

Naval Research Laboratory
Washington, DC 20375-5000

In Response to:

Contract #N00014-85-C-2444

12 May 1988

Contracted Deliverables

The following report is the final report on the prototype architecture for the demonstration summarization system as required in paragraph A002 of the RFP for contract N00014-86-C-2444. Also, included in this report is some documentation and instructions for the software. This, in addition to the documented software being delivered, constitutes the deliverable described in paragraph A003. The software is present on the Symbolics at NCARAI.

1. Final Report on the Architecture

1.1. The Original Text Reduction System

The initial prototype of the TERSE (TExt Reduction SystEm) system was developed around 1983 and is described in NRL Report 8893. This system was implemented in OPS5. The architecture of this system was very flat. In OPS5, there are no concepts of separate knowledge bases, rule classes or other means of internally organizing the information contained in these system. The only organization possible is for the user to explicitly order elements in his file and add comments for his own benefit.

This methodology was adopted in OPS5 because it was thought that rule systems should be completely *data driven*. Rules simply exist in the system, each unrelated to another until the data is introduced. At that point each individual rule attempts to collect enough data to instantiate all of its premise conditions. At all times, an overseeing control strategy is examining what rules have complete instantiation sets. If more than one rule is ready to fire at the same point in time, then the control strategy determines which rule will fire first, second, and so on. The control strategy employed in the original system (the MEA strategy) is particularly sensitive to the *age* of the fact which matches the first premise of a rule. The reader is referred to[1] for more information on the control strategy and OPS5, in general.

Every rule in the prototype system had a special first premise. This premise was what in effect, caused there to be several distinct phases of processing. Rules with the same first premise were conceptually related in the sense that they were working towards a similar goal. It did not matter in what order the rules which constituted this "pseudo-set" of rules fired with respect to one another.

The behavior of the rule system was not purely sequential. It was not the case that all the rules with the same first premise would fire before all the rules with another first premise. But it was the case that if the same set of facts was matched by two or more rules with different first premises, a definite order was imposed on the firing of the candidate rules.

1.2. The Demonstration System (TERSE)

The newest version of the demonstration system (TERSE) is implemented in KEE (Knowledge Engineering Environment). The system runs on both a Symbolics and a Sun workstation. All the code for the system is written Common Lisp and KEE. The prototype system contained many of the rules that were carried over to the demonstration system. However, another large knowledge base which includes a model of a *starting air compressor* and rules for using that model to help disambiguate nominals and to infer implicit causality was added to the demonstration system. Other than this new addition, the main difference between the prototype system and the demonstration system is the difference in architecture and the control of the rule system. These aspects of the demonstration system will be described below. The domain model portion of the system will not be discussed in this report. A NRL technical report has been written in which the domain model is described in great detail[2].

1.2.1. KEE (Knowledge Engineering Environment)

KEE provides the ability to maintain multiple knowledge bases. KEE also provides a variety of ways to represent knowledge in these knowledge bases. Built into KEE are both frames and rules. It is also possible to encode knowledge in the form of Common Lisp procedures.

The most basic unit in KEE is the frame structure which is called a **unit** in KEE terminology. The system is a complete frame system in terms of what is normally associated with frame systems. There are ways to associate frames to one another via links which form inheritance lines. Frames can either be subclasses of one another or they can be instances of a class. The slots of the frames can contain pointers to other frames, simple values, or procedures. Demons which fire when a value is accessed, added, or deleted can also be attached to slots.

In addition to the frame paradigm that KEE provides, it is also possible to treat frame units as objects, in the object-oriented sense. When a frame has a slot filled with a procedure (called a method in KEE), *messages* can be sent to that frame and if the frame contains the appropriate method, the message will be handled by the frame.

The rule system in KEE is actually implemented using the frame system. There are a variety of rule types to choose from: deduction rules, same world action rules, new world action rules. There are explicit forward chaining and backward chaining behaviors in the rule system. Because rules are implemented in the frame system, some characteristics of the frame system are carried over to the rule system. For example, rules can be arranged into classes and subclasses. Although, the actual rule premises and conclusions are not inherited, other aspects affecting the rule's behavior can be inherited. The reader is directed to the KEE manuals[3] for more information on this system.

1.2.2. Control Strategy

The control strategies in KEE differ from those offered in OPS5. Our discussion from this point on will only include facts relevant to *deduction rules* because this is the only type of rule we have employed in the demonstration system. None of the provided control strategies in KEE take into account the *age* of facts used to match a rule's premise. If two or more rules are matched at the same time, the default KEE strategy is to just push the two rules onto the forward chaining agenda in essentially arbitrary order relative to each other. If you look at the order in which rules fire and the consequent rules that are tickled after a rule fires as a tree, the default control strategy of KEE traverses this tree in a depth first fashion but nodes with the same parent are not ordered in any particular way.

There are other control strategies supplied in KEE. None of these strategies deal with the recency of facts because facts are not time-stamped as they are in OPS5. However, to most closely duplicate the way the prototype system works, it is possible to use rule weights in KEE. There are two corresponding control strategies which order rules based on their associated weights. However, we did not choose to implement the system in this way but felt that it should be mentioned as another possible organization of the system. (This approach would affect the organization of the system because it would necessitate the inclusion of the rules from the domain model into the same knowledge base as the other rules.)

In the current implementation, we have used only the default forward chaining control strategy. There are two separate knowledge bases in the current system. Each knowledge base contains several different rule classes. Since it is possible to invoke a particular class through a single statement, we have lisp control modules which invoke our rule classes in a specific order. Within a rule class invocation, the default control strategy determines how the rules will fire. Each of our rule classes corresponds roughly to the original prototype rules which had identical first premises. However, the KEE rules no longer contain a special first premise since we have used explicit control modules to control the order in which the rules fire.

1.2.3. TERSE: Reorganized

KEE is a more sophisticated system than OPS5. The conversion to KEE has allowed us to break up the rule system into more modular, separate knowledge bases. The new architecture consists of three main knowledge bases. The first knowledge base is called "datastructures". It contains all the datastructures used to define information formats received from the parser and it also contains all the control information for applying other knowledge bases. The second knowledge base is called "TERSE". It contains the original summary system rules. The third knowledge base contains the domain model (model of the SAC). Diagram 1 depicts the system architecture.

TERSE ARCHITECTURE

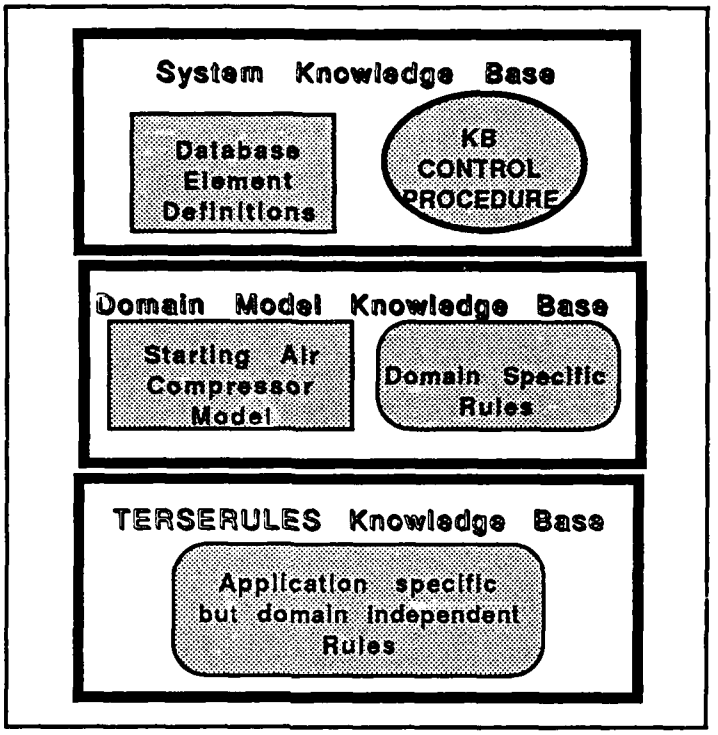


diagram 1

TERSE Data Flow

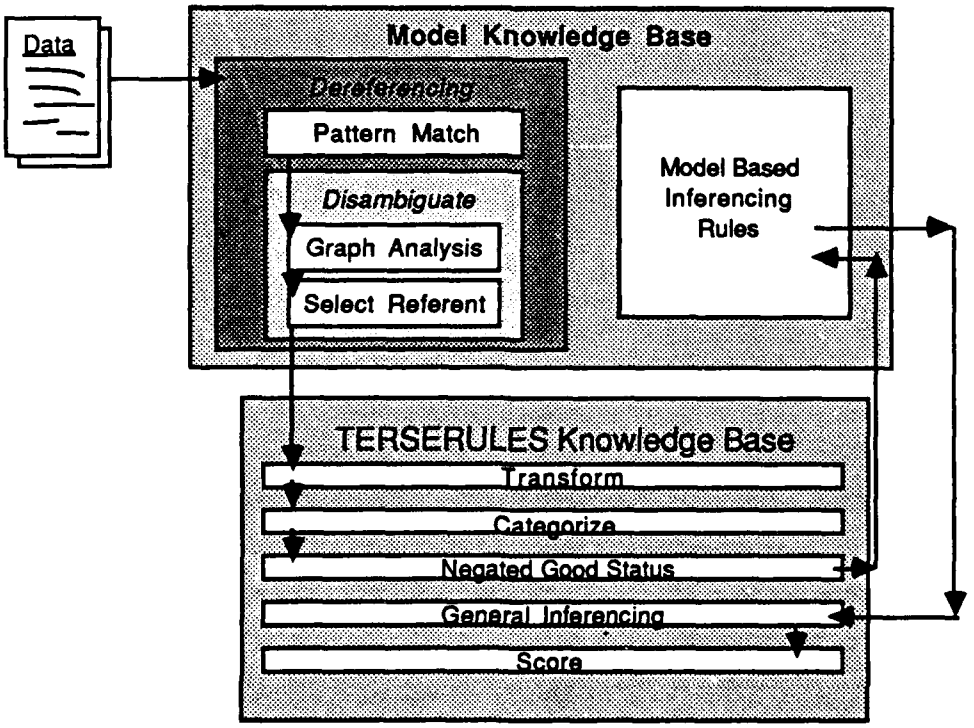


diagram 2

1.2.4. The Rule Classes

The Terse.rules class contains five subclasses: **Transform, Categ, Negated.good.status, General.infer**, and **Score**. These classes were divided up in this way to provide a better conceptual organization. Rules that are designed to work towards similar goals are grouped together in the same rule class.

The **Transform** class of rules performs operations which accomplish further normalization of causal structures. The objective is to change statements of the form "x is due to y" and "y impaired x" into the canonical form "y caused x". Removing this variability while retaining the causal implications of the original statement, simplifies subsequent rules because they do not have to look for the different variations.

Rules contained in the **Categ** class categorize elements contained in the statements. The categories represent concepts that are considered to be important in the application. In this application, the concepts of CAUSE, QUANTITY, MALFUNCTION, and DAMAGE are important. Certain quantities (zero in particular), MALFUNCTION and DAMAGE are all further categorized under a concept which represents BAD conditions. This process of categorization is done in order to segregate out the information that is provided by the author that is not directly expressing something about the failure.

The two rules contained in the **Negated.good.status** class simply check to see if something which was **not** categorized as BAD was modified by a negation or a zero quantity. If such an occurrence is found, the classifier of BAD is added.

General.infer contains only one rule which performs domain independent inferencing. This class was created to help separate domain specific inferencing rules from domain independent rules.

The **Score** class contains all the rules that assign a score to anything in the information formats. This scores assigned in these rules reflect what concepts are considered important in this application.

1.2.5. Rule System Control

Diagram 2 on page 5 shows the possible flow of control of the TERSE system. There are two subcomponents of the MODEL knowledge base. The first subcomponent handles dereferencing of part nominals in the messages. Once the parts have been identified, control is then passed to the TERSE knowledge base. The control module (contained in the DATASTRUCTURES knowledge base) passes control from one rule class to another in a sequential manner. This application operates strictly in forward-chaining mode. The order in which *tickled* rules fire, is determined by the default mechanisms provided by KEE. Ties are also broken with the default method provided by the KEE system.

2. Instructions for TERSE

The following document is a set of step-by-step instructions for running the TERSE system. These instructions should enable someone who is able to perform minimal functions on the Symbolics and in KEE to run the system and process messages successfully.

2.1. Booting the Symbolics

The Symbolics should be booted up with the KEE world. This world is currently named "g-kee.boot". By typing in "boot g-kee.boot" the system will bring up KEE. Once the Symbolics is finished booting, a login message will appear. The user should then log by typing *login NAME* where NAME is an appropriate user i.d. Once logged in, KEE can be accessed by typing **select** *k*". The KEE display will appear in a few minutes. When the screen inverts (briefly), this means that KEE is ready to go.

2.2. Loading the Knowledge Bases

In order to load the knowledge bases, KEE should be booted up and preferably have no other knowledge base loaded. The "current default directory" should be the directory where all the TERSE knowledge bases are stored. To load the knowledge bases, mouse on the key icon and mouse on the "load KB" menu option. A prompt will appear in the typescript window requesting the name of the knowledge base to be loaded. Enter *datastructures.u*. Loading this knowledge base into KEE will cause all other required knowledge bases to be loaded. This can take up to 10 minutes to load. Towards the end of the loading process, the mouse icon will appear and the user will be asked if he/she wishs to create the viewports or images for certain knowledge bases. You should always indicate "yes" by clicking the leftmost mouse button. Answer positively for all the questions that may appear, regarding viewports or images during this loading process.

2.3. Getting to the System

Once the system is loaded, an image panel with a ship and a computer terminal will appear on the screen. This is simply the introductory panel and you can move into the TERSE system by left mousing on the icon picture of the computer terminal. This will cause this image panel to disappear and other panels to appear.

2.4. The Command Panel

The command panel is located in the upper left-hand corner of the screen. This panel contains four different icons representing the available functions in the TERSE system. The "glasses" icon represents the function of **reading** in a new message from a file. The "computer" icon represents the function of **running** the system on the current message. The "pencil" icon represents the function of **erasing or deleting** the current message from the KEE knowledge bases (not from the actual file). The "book" icon, when clicked on, will perform the function of **flipping to the next page** of message formats if they all do not fit on one page. If you click on this icon and exactly the same display reappears, then there was only 1 page of displays. You can tell how many pages there are for a message by looking in the top line of the format display panel. (This is only visible AFTER you have loaded in a message.) A message indicating what page of how many is displayed in the title bar. If for some reason you should want to bring back the

original introductory panel, "little ship" icon can be clicked on and the panel will reappear. This has no real effect on the system, and can be done if you wish to leave the system with "a pretty picture" on the screen.

2.5. Reading In a Message

To read in a message, left click on the glasses icon. You should see the icon become highlighted and then a prompt will appear in the typescript window. The prompt will be requesting the name of the file containing the message to be read in. The user should type the name of the file including the correct path if the message is not located in the default directory. Once the user has entered the file name, the system will begin reading in the message. The listified message is echoed back in the typescript window. After that, the screen will be covered with a blank panel. Shortly, boxes representing the formats will begin appearing on the panel. Some boxes will appear and then disappear. This is normal. It is also normal that the boxes appear in the reverse order form the order in which they were read.

2.6. Running the System

Once all the format boxes have been displayed, the glasses icon will return to normal (become un-highlighted). At this point, if you wish to have the rule system run, you should left click on the "computer" icon. While the system is running, messages are printed out on the typescript window. This window will pop up over the format's display panel. It is necessary to left click on the format display panel when the system is finished running to get that window to fall to the background and also to get the display panel to update its display and show which format has been chosen. This will be evident because the chosen format(s) will be highlighted. It can take around five minutes for the system to run. Most of the time-consuming processing is in the nominal dereferencing because a large search space is being considered.

2.7. Examining the Results

Once the system has finished running, the user can look at explanation boxes attached to each format. These boxes contain English descriptions of the rules that effected the score of the format. The explanation box for every format does not fit on the screen. The user can view one explanation box at a time by left clicking on the box containing the format's text. A new box will appear on the right hand side of the screen. This is the explanation box. Sometimes, so many rules fire that all the explanations do not show up in the box. This box is a "text display" from the supplied KEE pictures knowledge base. By right clicking, a menu that contains options such as scrolling the text will appear. If the user left clicks on the box, it will disappear again. It can be retrieved by clicking on the format text box again.

2.8. Processing Another Message

If you wish to process another message, you can click on the glasses icon again. The system will delete the current message (if there is one) and ask you for the file name of a new message. Once a new message is read in again, the user can click on the "computer" icon to process the message.

2.9. Deleting a Message

If you just want to delete the current message without reading in a new one, click on the "pencil" icon. This removes the message from the current knowledge bases.

2.10. Shutting Down the System

It is not necessary to do any "housekeeping" in order to reboot the system. Just retrieve the main lisp listener window and type "stop machine". This will log you out and get back to FEP.

2.11. Abbreviated Instructions

All the screen dumps referred to in the following section, show the state of the display **after** the corresponding step is executed.

1 type *boot g-kee.boot* (boot the KEE world)
diagram 3

2 type *login USER* (log in with your user name)
diagram 4 shows screen after step 2

3 type **select** *k* (brings up KEE screen)
diagram 5

4 **mouse** on "key" icon

5 choose *load KB* menu option

6 respond *datastructures.u to prompt for name* of knowledge base

7 respond yes by left clicking to all questions during load
display 6

8 **mouse** on "terminal icon" to start TERSE

9 **mouse** on "glasses icon" to read in a message
display 7

10 **mouse** on "computer icon" to process message
display 8

11 **mouse** on a format text box to see explanation box
display 9

12 **mouse** on "pencil icon" to delete the current message

12 type *stop machine* (in lisp listener to prepare to reboot machine)

13 type *boot* (boot.file) (boot using either g-pcl.boot or g-kee.boot)

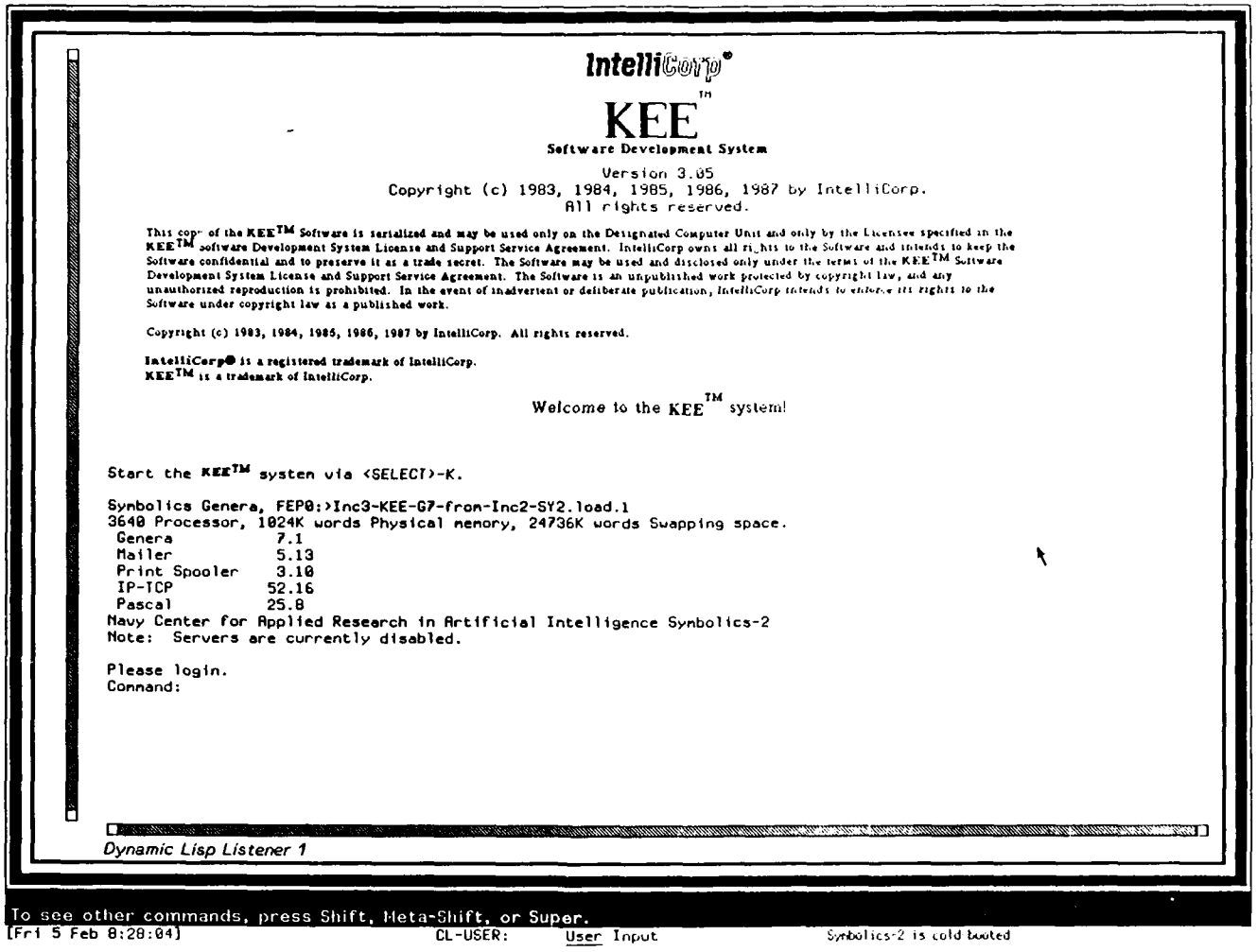

IntelliCorp
KEE
Software Development System
Version 3.05
Copyright (c) 1983, 1984, 1985, 1986, 1987 by IntelliCorp.
All rights reserved.
This copy of the KEE Software is serialized and may be used only on the Designated Computer Unit and only by the Licensee specified in the KEE software Development System License and Support Service Agreement. IntelliCorp owns all rights to the Software and intends to keep the Software confidential and to preserve it as a trade secret. The Software may be used and disclosed only under the terms of the KEE Software Development System License and Support Service Agreement. The Software is an unpublished work protected by copyright law, and any unauthorized reproduction is prohibited. In the event of inadvertent or deliberate publication, IntelliCorp intends to enforce its rights to the Software under copyright law as a published work.
Copyright (c) 1983, 1984, 1985, 1986, 1987 by IntelliCorp. All rights reserved.
IntelliCorp is a registered trademark of IntelliCorp.
KEE is a trademark of IntelliCorp.
Welcome to the KEE system!
Start the KEE system via <SELECT>-K.
Symbolics Genera, FEP0:>Inc3-KEE-G7-from-Inc2-SY2.load.1
3640 Processor, 1024K words Physical memory, 24736K words Swapping space.
Genera 7.1
Mailer 5.13
Print Spooler 3.10
IP-TCP 52.16
Pascal 25.8
Navy Center for Applied Research in Artificial Intelligence Symbolics-2
Note: Servers are currently disabled.
Please login.
Command:
Dynamic Lisp Listener 1
To see other commands, press Shift, Meta-Shift, or Super.
[Fri 5 Feb 8:28:04]
CL-USER:
User Input
Symbolics-2 is cold booted

diagram 3

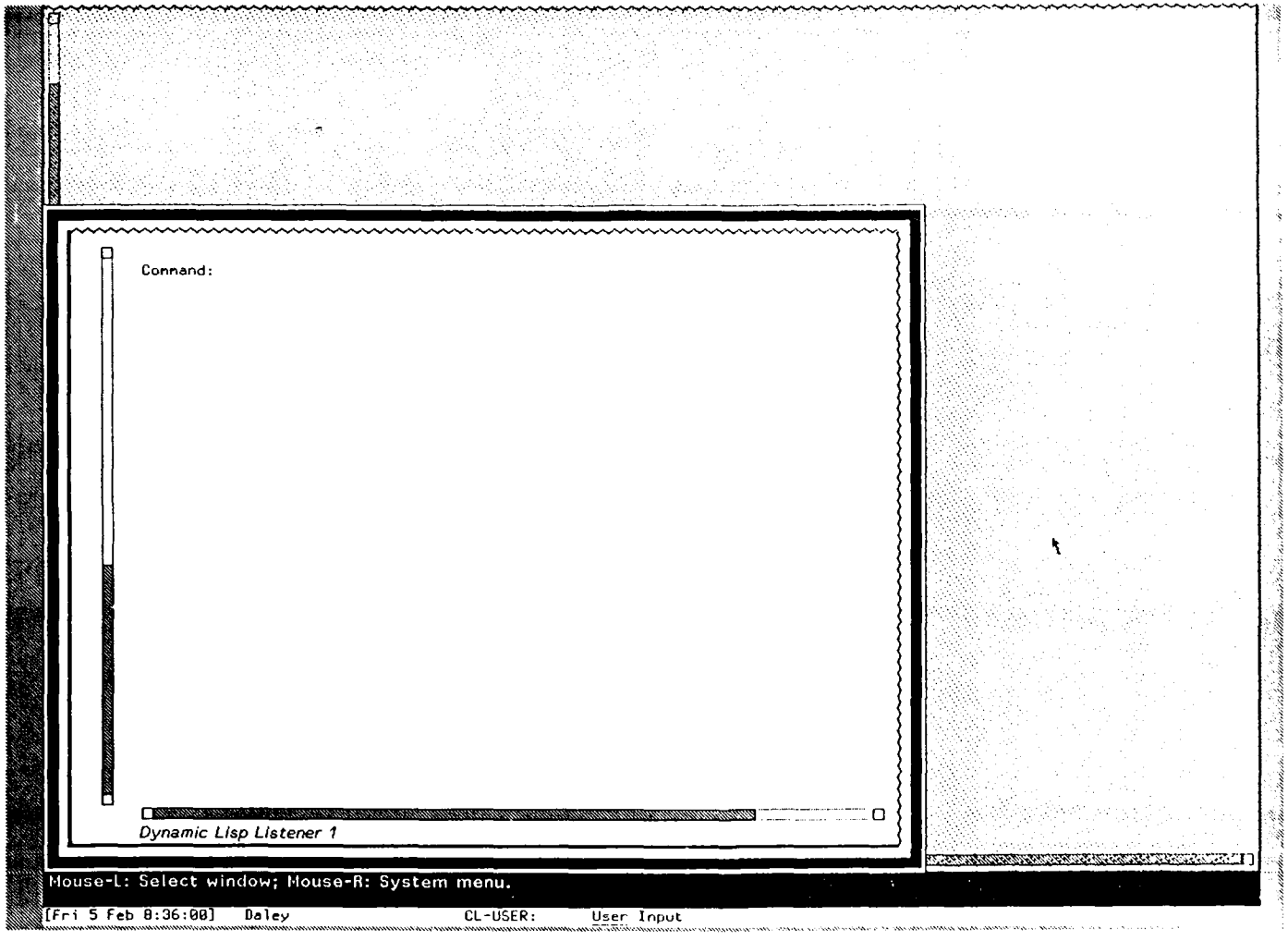

Command:
Dynamic Lisp Listener 1
Mouse-L: Select window; Mouse-R: System menu.
[Fri 5 Feb 8:36:00] Daley
CL-USER:
User Input

diagram 4

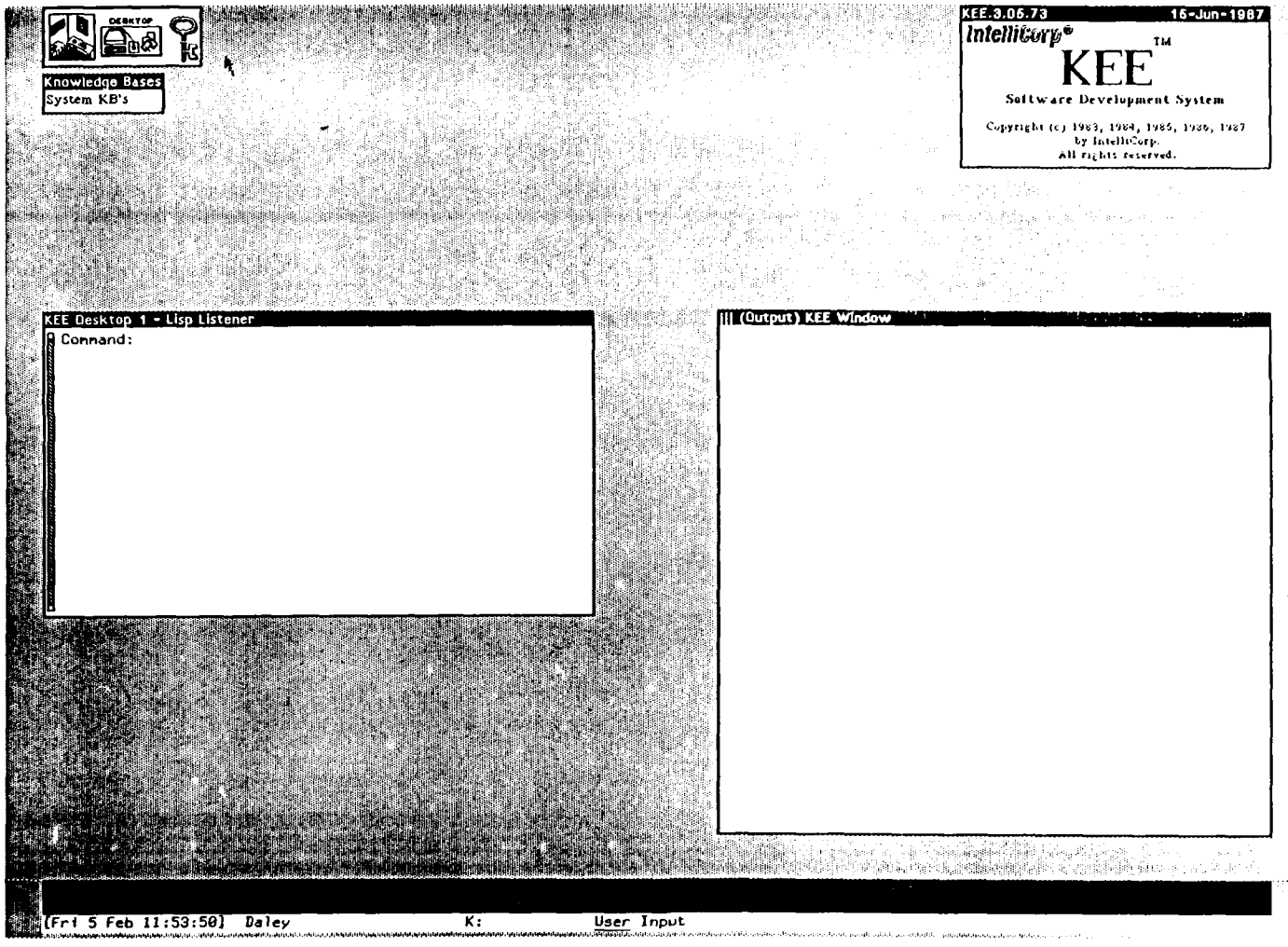

DESKTOP
Knowledge Bases
System KB's
KEE.3.06.73
16-Jun-1987
IntelliCorp
KEE
Software Development System
Copyright (c) 1983, 1984, 1985, 1986, 1987
by IntelliCorp.
All rights reserved.
KEE Desktop 1 - Lisp Listener
Command:
(Output) KEE Window
(Fri 5 Feb 11:53:50) Daley
K:
User Input

diagram 5

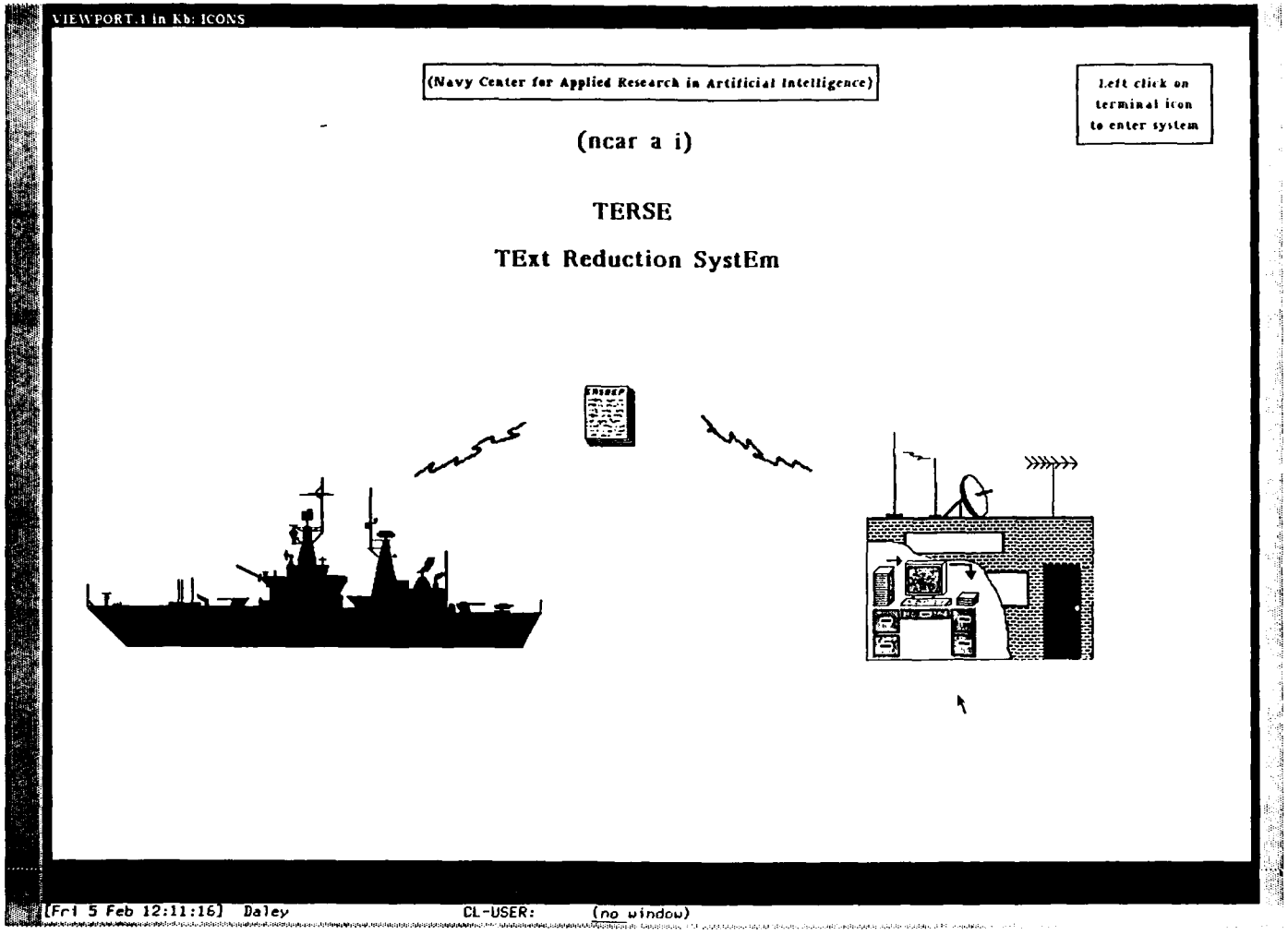

VIEWPORT.1 in Kb: ICONS
(Navy Center for Applied Research in Artificial Intelligence)
Left click on terminal icon to enter system
(ncar a i)
TERSE
TExt Reduction SystEm
(Fri 5 Feb 12:11:16) Daley
CL-USER:
(no window)

diagram 6

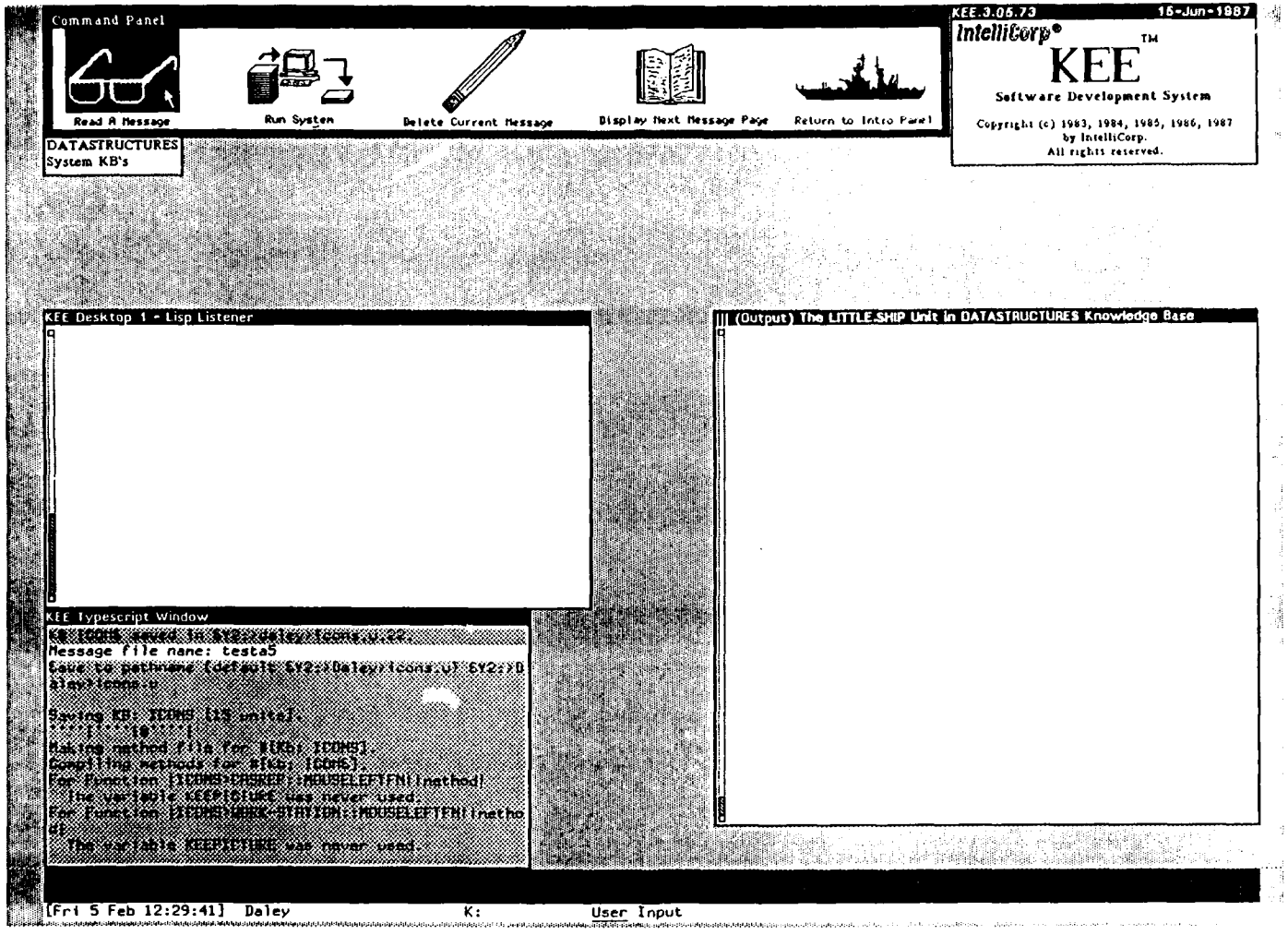

Command Panel
Read A Message
Run System
Delete Current Message
Display Next Message Page
Return to Intro Panel
KEE.3.06.73
15-Jun-1987
IntelliCorp®
KEE
Software Development System
Copyright (c) 1983, 1984, 1985, 1986, 1987
by IntelliCorp.
All rights reserved.
DATASTRUCTURES
System KB's
KEE Desktop 1 - Lisp Listener
(Output) The LITTLE.SHIP Unit in DATASTRUCTURES Knowledge Base
KEE Typescript Window
Message file name: testa5
Saving KB: ICONS [13 units].
Making method file for #[KB: ICONS].
Compiling methods for #[KB: ICONS].
The variable KEEPICTURE was never used.
The variable KEEPICTURE was never used.
[Fri 5 Feb 12:29:41] Daley
K:
User Input

diagram 7

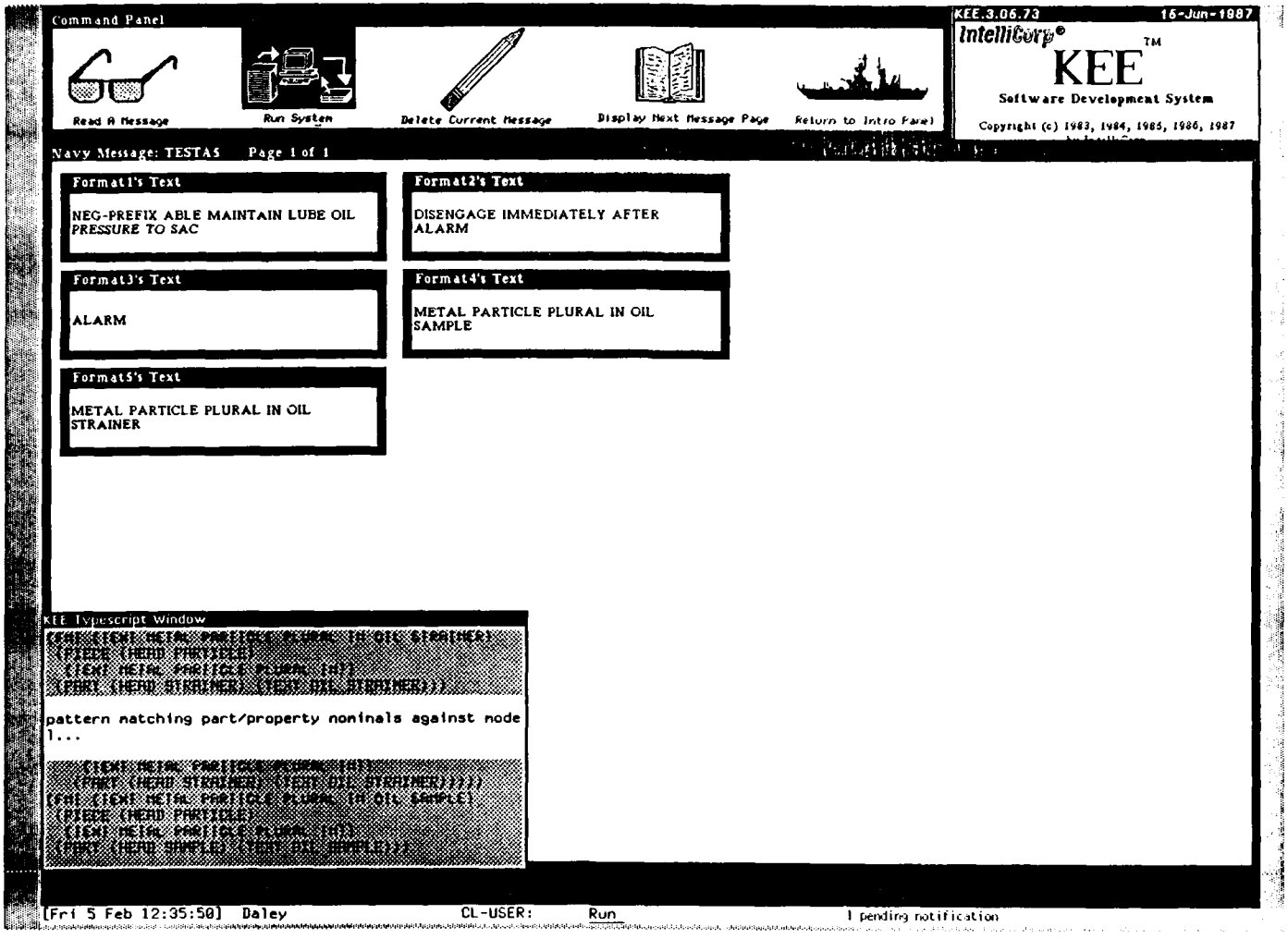

Command Panel
KEE.3.06.73
15-Jun-1987
IntelliCorp
KEE
Software Development System
Copyright (c) 1983, 1984, 1985, 1986, 1987
Read A Message
Run System
Delete Current Message
Display Next Message Page
Return to Intro Panel
Navy Message: TESTA5 Page 1 of 1
Format1's Text
NEG-PREFIX ABLE MAINTAIN LUBE OIL PRESSURE TO SAC
Format2's Text
DISENGAGE IMMEDIATELY AFTER ALARM
Format3's Text
ALARM
Format4's Text
METAL PARTICLE PLURAL IN OIL SAMPLE
Format5's Text
METAL PARTICLE PLURAL IN OIL STRAINER
KEE Typescript Window
pattern matching part/property nominals against node
1...
[Fri 5 Feb 12:35:50] Daley
CL-USER:
Run
1 pending notification

diagram 8

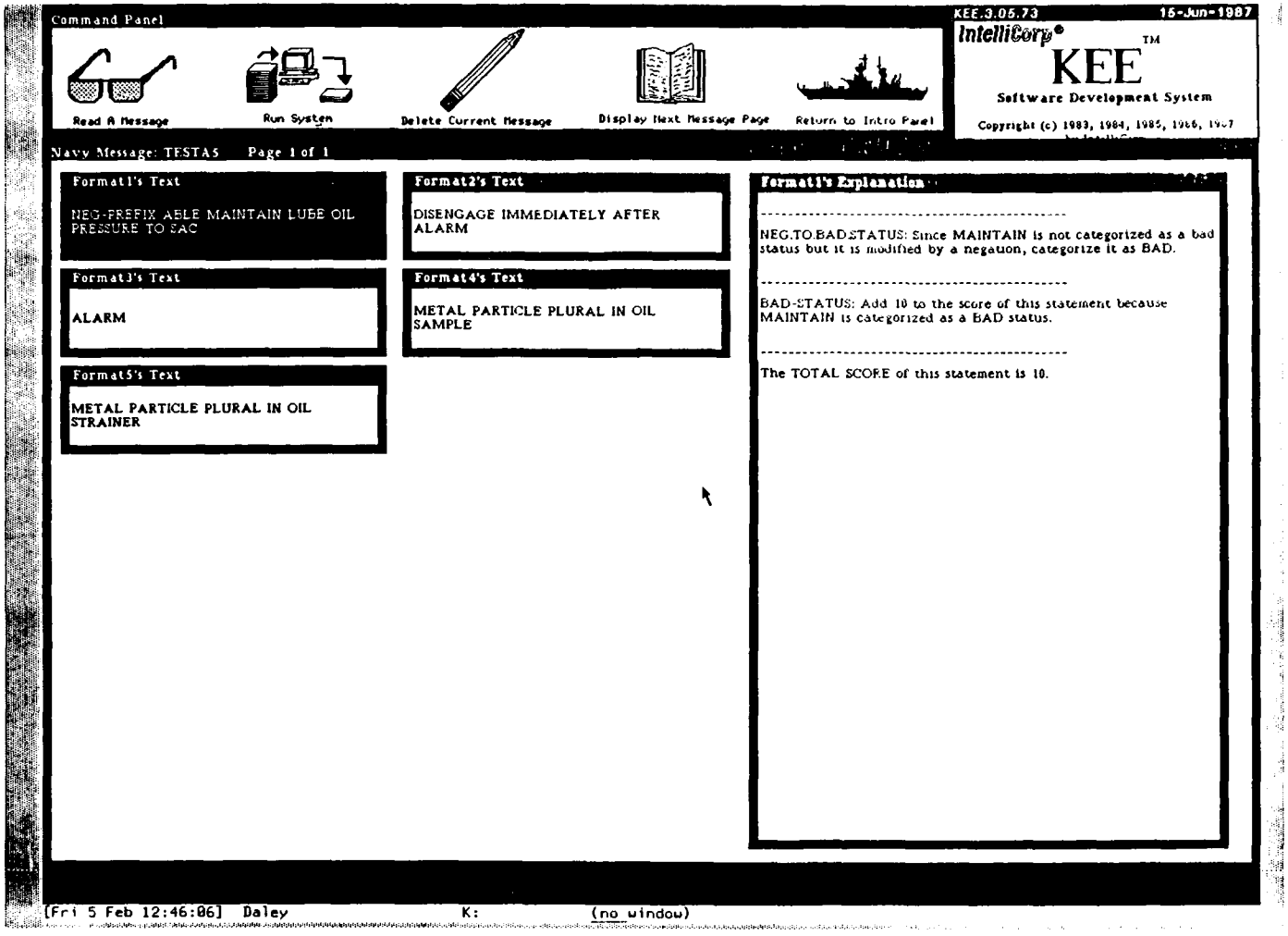

Command Panel
KEE.3.05.73
16-Jun-1987
IntelliCorp
KEE
Software Development System
Copyright (c) 1983, 1984, 1985, 1986, 1987
Read A Message
Run System
Delete Current Message
Display Next Message Page
Return to Intro Panel
Navy Message: TESTA5 Page 1 of 1
Format1's Text
NEG-FREFIX ABLE MAINTAIN LUBE OIL PRESSURE TO SAC
Format2's Text
DISENGAGE IMMEDIATELY AFTER ALARM
Format3's Text
ALARM
Format4's Text
METAL PARTICLE PLURAL IN OIL SAMPLE
Format5's Text
METAL PARTICLE PLURAL IN OIL STRAINER
Format1's Explanation
NEG.TO.BAD.STATUS: Since MAINTAIN is not categorized as a bad status but it is modified by a negation, categorize it as BAD.
BAD-STATUS: Add 10 to the score of this statement because MAINTAIN is categorized as a BAD status.
The TOTAL SCORE of this statement is 10.
(Fri 5 Feb 12:46:06) Daley
K:
(no window)

diagram 9

2.12. Miscellaneous Features (Not for Naive Users)

In order to run the system over again without deleting and rereading a message, you can run the method "undo-rule-effects". This method is attached to the "terse.system" unit in the "datastructures" KB. It basically performs housekeeping and fixes things back to the state they were in before the message was processed. After running the "undo-rule-effects" method, just click on the "computer" icon again to run the rule system as usual.

On the initial display panel, if you click on the "casrep" icon when a message is already loaded up, a graphical tree display of the message will appear. This graph is not all that great because it has to ignore a lot of stuff in the formats in order to make it even partially legible. In order to get rid of the window containing the tree, press "shift right click" and choose the "bury" menu option. If you "kill" this window, you will not be able to display another tree until rebooting the system. This option was built in strictly for demonstration purposes and it isn't really that useful.

References

1. Lee Brownston, Robert Farrell, and Elaine Kant, *Programming Expert Systems in OPS5,* Addison-Wesley Publishing Company, Inc., Reading, MA, 1985.
2. K. Wauchope, M. K. DiBenigno, and E. Marsh, "Automated Text Highlighting of Navy Equipmnet Failure Messages," *submitted as a NRL Technical Report,* Washington, D.C., April, 1988.
3. *KEE User Reference Manual,* Intellicorp, Inc., San Diego, CA, 1988.

END

DATED

FILM

9-88

DTIC